Algorithm interpreter

Project 3

EFREI International Section

By Thibault LEPEZ & Camille BRIAND

1. Project goals

The main purpose of the project was to create an algorithm interpreter able to debug line by line in the console and to take inputs files. The subject gave us some bonus features as the support of floating numbers or error handling. We wanted to add as much functions as possible and we gave ourselves the challenge to be very memory efficient.

The memory management was one of our main concern. We constantly check memory leaks and memory allocation. Right now, our program is memory leak free and we are proud of that.

To have a rigorous code the organization of the work was very important I our project.

1. Work organization

In the project 2 we used Dropbox to work together and even if it’s a very efficient way to exchange files the main problem was that we could not work on the same file without resulting in conflict when saving our work. Another problem with Dropbox was that our saves were the same so any mistake instantly uploaded in all of our computers.

Therefore, after knowing about GitHub we decided to change our organization.

We learned how to use GitHub in our association Efrei-Linux. GitHub allowed us to upload our codes only if we were sure of it’s quality. It also allowed us to work on the same file as long as we don’t change the same part, which is much more convenient.

The only “problem”, even if it’s not really one, was that GitHub is much more easy and fast to use on Linux.

1. Programming and issues

We started the project directly after knowing about the subject. But at first, we thought it was mainly an algorithm translator into C language. After reading the subject we understood that it was an algorithm interpreter, as we both did some python programming we were inspired by the native python interpreter.

On firsts weeks, we started programming in Windows OS as the major part of the class did so. But even if we were working with Windows we implemented the possibility to choose between UNIX based OS and WINDOWS.

By working with Windows, we started the project with Dropbox. But during the project we learn about GitHub and this made us change our organization and we decided to work with LINUX.

When we compute our program the first time with LINUX it instantly crashes for memory reasons. At this moment, we understood that Windows was much more permissive with memory as we had no problems since then. We opened an issue in our GitHub to keep the process of our program (link: <https://github.com/Mr-Monster-0248/C-project-3/issues/1> ).

Thanks to Mr. Klai who advised us to use Valgrind we manage to fix our memories issues. Valgrind took us about a week to understand how to use it. And finally, it became indispensable for our program as we wanted to have the more efficient one.

After solving memory issues, we started the calculus function. We wanted it to respect priority order and to understand parenthesis. To do so we use many functions, the first one to find the highest priority operator and which return the position of the character. Then another function compute the operation between the numbers. And we do so while there is only one number left. Adding the parenthesis support was challenging. At first we had memory issues with the recursive call of the function (GitHub link: <https://github.com/Mr-Monster-0248/C-project-3/issues/3> ).

We added the string concatenation very easily as we stock all our variables with strings. We just needed to recognize when to compute the strings operations.

The support of the boolean expression was built as the operation function. Once we recognize a comparison we compared both numbers or expression. The only problem is that we are not able for the moment to compute an operation inside a comparison as with (2 + 2 = 4). But maybe by changing the recognition of an operation we should be able to compute this sort of expression.

Our next challenge was to support variables, to do so we used structures. More precisely we use an array of structures to stock those variables. For the project defense, we weren’t able to use those variable but the stocking was working. Finally, we manage to use variables

1. Conclusion

This project thought us a lot and was very interesting. We will probably continue the project after the rendering date. As we put it on GitHub with this link <https://github.com/Mr-Monster-0248/C-project-3>